VISIONS FOR THE LINUX KERNEL PWM SUBSYSTEM

ABOUT ME

- Senior Software R&D Engineer @ Baylibre
- PWM subsystem maintainer
- contributor to various kernel subsystems
- ukleinek on libera and OFTC
- PGP: E2DCDD9132669BD6

... AND BAYLIBRE

- Embedded software consultancy based in Nice, France
- ~60 engineers
- various OSS projects (Linux, U-Boot, Zephyr, ...)

Baylibre

MOTIVATION FOR THIS TALK

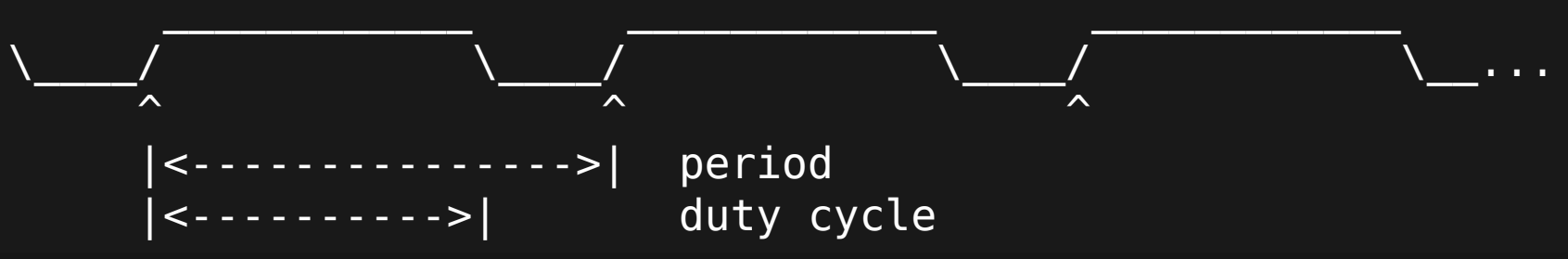
- Present ideas for improvement (and pending changes)
- Who uses PWMs for what?
- What are your pain points and work arounds?

WHAT IS A PWM?

- periodic square wave signal
- used to
  - blink or dim LEDs
  - drive display backlights
  - motor control (e.g. fan)
  - remote controls

WHAT IS A PWM?

- period + duty cycle [ns]
- polarity (normal or inverted)
- enable & disable



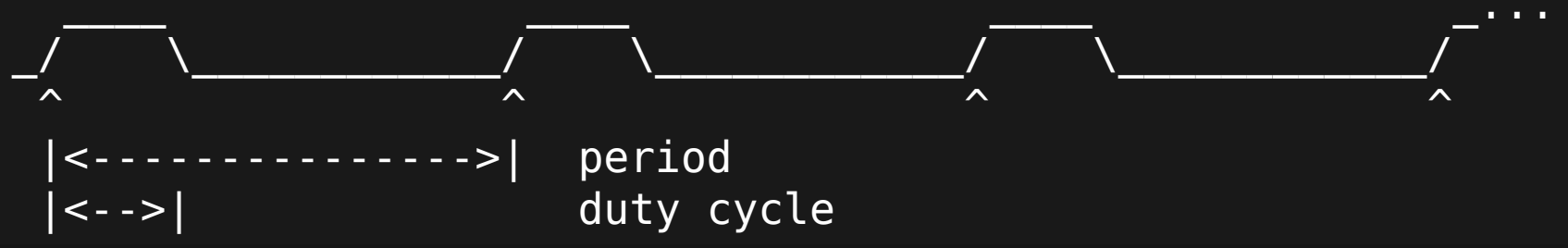
Baylibre

PWM SUBSYSTEM STATUS QUO

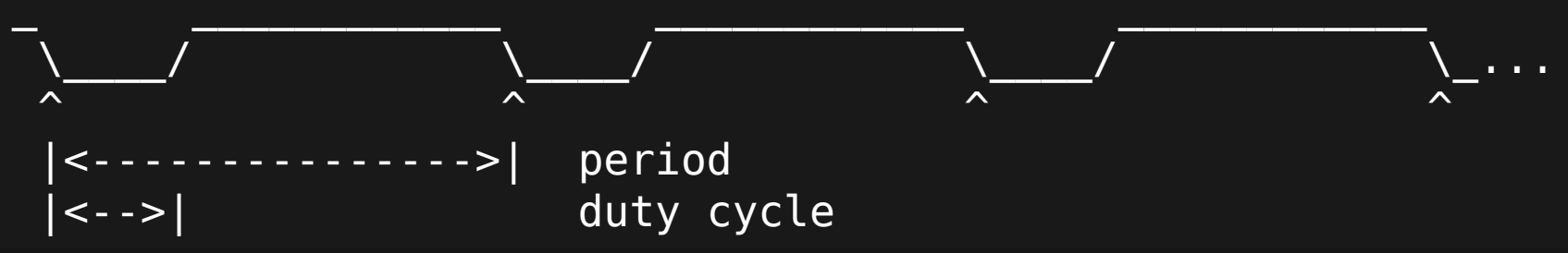
- period + duty_cycle + polarity
- sysfs userspace API

POLARITY

- normal polarity



- inversed polarity



SYSFS USERSPACE API

Current status quo:

```
# cd /sys/class/pwm
# pwmchip=0
# hwpwm=3
# echo $hwpwm > pwmchip$pwmchip/export
# echo 1000000 > pwmchip$pwmchip/pwm$hwpwm/period
# echo 500000 > pwmchip$pwmchip/pwm$hwpwm/duty_cycle
# echo inversed > pwmchip$pwmchip/pwm$hwpwm/polarity
# echo 1 > pwmchip$pwmchip/pwm$hwpwm/enable
```

SYSFS USERSPACE API: CHALLENGES

- configuration changes are not atomic
- intermediate states must be valid
  - period ≥ duty_cycle
  - subject to hardware limitations of used pwmchip (somewhat softend by v6.11-rc1, commit 9dd42d019e63)
- no way to determine current configuration
- no way to determine resulting configuration of a given request

LOWLEVEL CALLBACKS TODAY

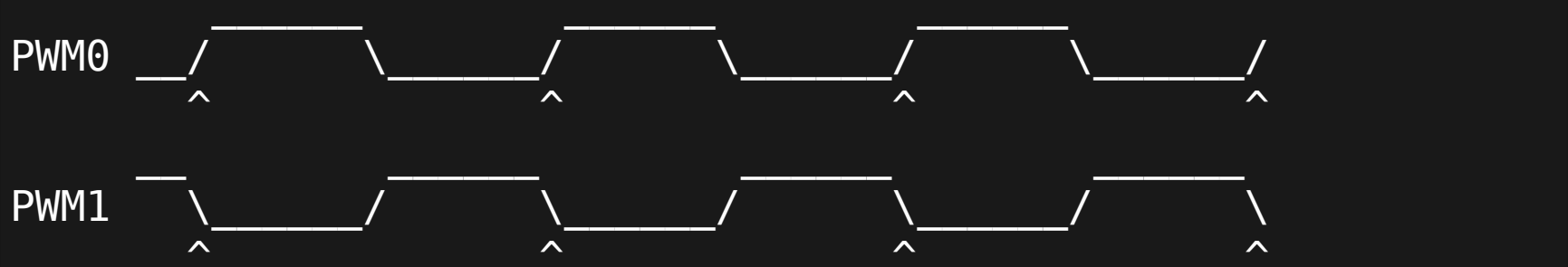
```
struct pwm_state {
        u64 period;
        u64 duty_cycle;
        enum pwm_polarity polarity;
        bool enabled;
        ...
};

int (*apply)(struct pwm_chip *chip,
             struct pwm_device *pwm,
             const struct pwm_state *state);
int (*get_state)(struct pwm_chip *chip,
                 struct pwm_device *pwm,
                 struct pwm_state *state);
```

IDEAS FOR THE PWM SUBSYSTEM

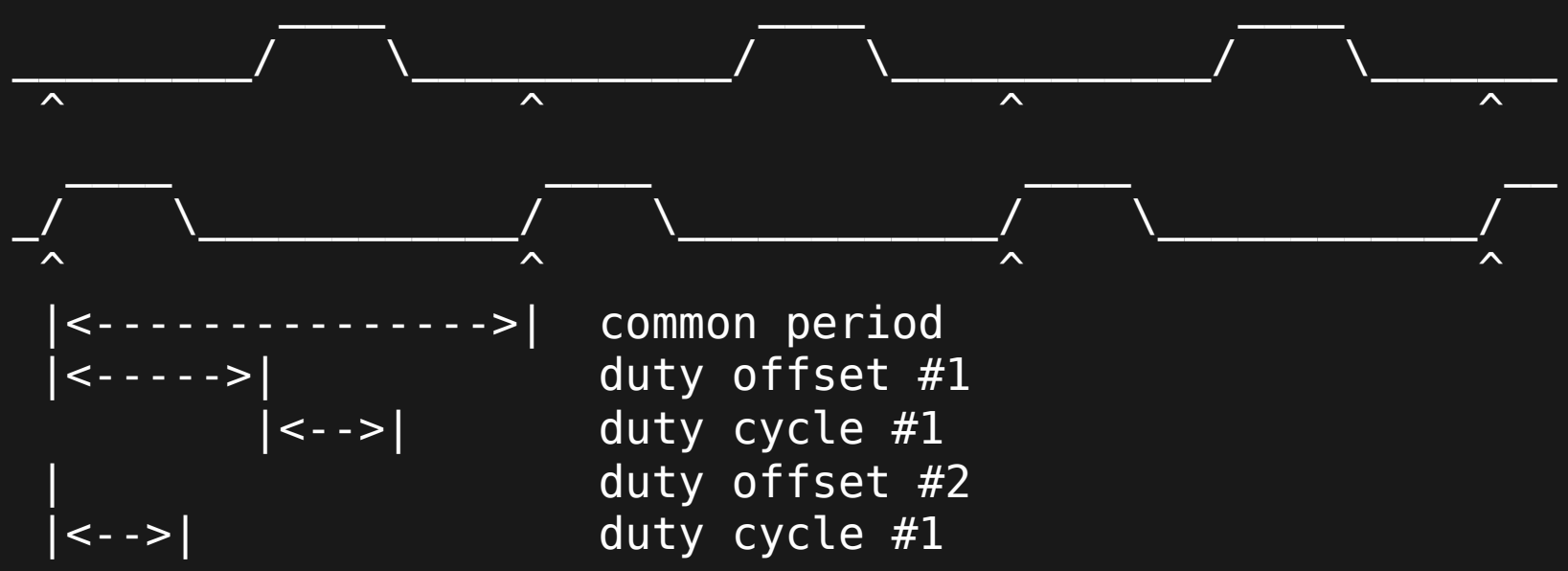
- coupling of channels
- duty offset instead of polarity
- stricter rules for lowlevel drivers
- querying hardware
- pico second support

COUPLING OF CHANNELS



DUTY OFFSET

DUTY OFFSET + COUPLED CHANNELS



EXACTNESS AND EXPECTED RESULTS

Configuration has to be adapted to match the hardware.

Query actually implemented configuration (before it happens?)

PICOSECOND SUPPORT

Is it needed?

Currently all PWMs I know have a quantum > 1 ns.

NEW WAVEFORM API + CHARDEV USERSPACE API

- atomic
- Exactness and expected result
- ~~channel coupling~~
- duty offset
- ~~pico second~~

CHARDEV USERSPACE API

A device per pwm chip (/dev/pwmchipX)

```
struct pwmchip_waveform {
        __u32 hwpwm;
        __u64 period_length_ns;
        __u64 duty_length_ns;
        __u64 duty_offset_ns;
};

#define PWM_IOCTL_ROUNDWF       _IOWR(..., struct pwmchip_wave
#define PWM_IOCTL_GETWF         _IOWR(..., struct pwmchip_wave
#define PWM_IOCTL_SETROUNDEDWF  _IOW(..., struct pwmchip_wavef
#define PWM_IOCTL_SETEXACTWF    _IOW(..., struct pwmchip_wavef
```

LOWLEVEL CALLBACKS IN THE FUTURE

```
int (*round_waveform_tohw)(struct pwm_chip *chip, stru
                           const struct pwm_waveform *
int (*round_waveform_fromhw)(struct pwm_chip *chip, st
                             const void *wfhw, struct
int (*read_waveform)(struct pwm_chip *chip, struct pwm
                     void *wfhw);
int (*write_waveform)(struct pwm_chip *chip, struct pw
                      const void *wfhw);
```

LOWLEVEL CALLBACKS STICKED TOGETHER

FINAL

- Does this improve things for you?
- What are you still missing?
- Questions? Suggestions?